Technical Cybersecurity

You look smashing!

Grows this way!

Buffer Overflows

WHAT'S THE PLAN?

- Create a vulnerable program
- Exploit vulnerable program
- Profit! (well, not so much)

Stack Frame B

Stack Frame A

x1: int # stored in a word

Local Variables for y1: int # stored in a word functionB(.)

z1: int # stored in a word

Base Pointer # stored in a word

Return Address # stored in a word

Function Arguments for 1: int # stored in a word functionB(.)

c1: int # stored in a word

x: int # stored in a word

Local Variables for

y: int # stored in a word functionA(.)

z: int # stored in a word

Base Pointer # stored in a word

Return Address # stored in a word

b: Function Arguments for

c: int # storfunctionA(.)

d: int # stored in a word

Smash Frame Main Frame

char buffer[BUF_SIZE]

Base Pointer # stored in a word

Return Address # stored in a word

Grows this way!

char* arg

Locals

Base Pointer # stored in a word

Return Address # stored in a word

Args

Buffer Overflows

WHAT'S THE PLAN?

- Create a vulnerable program
- Exploit vulnerable program
- Profit! (well, not so much)

Buffer Overflows

WHAT'S THE PLAN?

- Create a vulnerable program
- Exploit vulnerable program
- Profit! (well, not so much)

Grows this way! Smash Frame char buffer[BUF_SIZE] Return Address # stored in a word char* arg Locals Main Frame Base Pointer # stored in a word Return Address # stored in a word **Args**

Command Line Values

COMMAND LINE ARGUMENTS

- We'll submit the data via the program command line
- Read it from the argv[] argument vector
- Pass it into the smash(.) function
- Copy it into a local buffer

Tools

GDB (for debugging and core files), python

Let's get to it.